WT8601M02 使用说明书

目　录

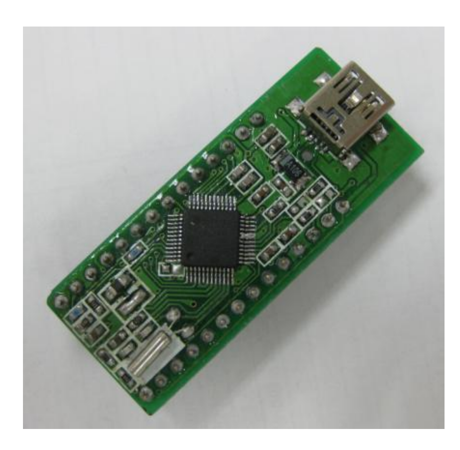
1、产品特性

- 支持 8Kbps～320Kbps 位速率 MP3 格式音频播放；
- 支持 VBR 可变位速率及 CBR 恒定位速率；
- 立体声高品质音频输出；
- 支持上电自动复位，低电压自动复位，外部管脚复位；
- 支持 MP3 控制模式、按键一对一控制模式、随机播放模式、DSA 控制模式；
- 具有上电自动播放、单曲循环、全部循环等功能；
- 采用 NAND-Flash 作为存储中心，存储空间大，语音时间长；
- 最大可支持 2GB byte NAND-Flash；
- 全速 USB2.0 数据传送；
- 低功耗运行，续航时间更长；
- 工作电压 DC5V。

2、应用范围

WT8601M02 能应用在高级家用电器，如智能语音导航电冰箱、语音导航空调、语音导航电磁炉等，以及高级玩具、汽车电子系统、长时间放音系统等高音质要求场所。

3、接线图示



WT8601M02 正面实物图片

1	NC	USB+5V	28
2	NC	D+	27
3	NC	D-	26
4	NC	GND	25
5	NC	P00	24
6	NC	P05	23
7	/RESET	VDD	22
8	VOUTRHP	NC	21
9	VOUTLHP	NC	20
10	GBUF	P01	19
11	DATA	P02	18
12	ACK	P03	17
13	STB	P04	16
14	GND	USBDET	15

WT8601M02 正面框图

WT8601M02 框图接线说明

管脚序号	功能	说明	管脚序号	功能	说明
1	NC	空	15	USBDET	USBDET
2	NC	空	16	P04	按键 4
3	NC	空	17	P03	按键 3
4	NC	空	18	P02	按键 2
5	NC	空	19	P01	按键 1
6	NC	空	20	NC	空
7	/RESET	复位脚	21	NC	空
8	VOUTRHP	音频 R 声道输出	22	VDD	电源正极
9	VOUTLHP	音频 L 声道输出	23	P05	按键 5
10	GBUF	音频地	24	P00	BUSY 输出
11	DATA	DSA 传送数据及开始标志	25	GND	USB 电源地
12	ACK	DSA 传送应答标志	26	D-	USB_D-
13	STB	DSA 数据保存标志	27	D+	USB_D+
14	GND	电源地	28	USB+5V	USB 电源

注： 1、GBUF 不能接电源地。 2、USBDET 为 USB 电压电压检测，模块内部已有处理电路，可以不接。

4、电气参数

环境温度：25℃ 输入电压 DC5V P00 上拉电阻：10KΩ

参数	标记	环境条件	最小值	典型值	最大值	单位
工作电压	V_{DD}	F_{sys}=12MHz	3.5	5.0	6.0	V
工作电流 1	I_{OP1}	没有负载	24.2	26.5	28.0	mA
工作电流 2	I_{OP2}	R_{ROUT}=8Ω R_{LOUT}=8Ω	26.0	27.5	46.6	mA
停止电流	I_{DD2}	没有负载	---	12.0	---	mA
上拉阻	RH	RESETn	---	75	---	KΩ

5、功能设置

当前 WT8601M02 可以设置以下功能。

序号	工作方式	详细操作	描述
1	语音循环方式	(1)、单曲语音循环	当前选中的语音一直循环播放
		(2)、全部语音循环	所有存储区里面的语音循环播放
		(3)、单曲播放停止	播放完当前的语音即停止
2	控制模式	(1)、MP3 控制模式	有播放/暂停、停止、上一曲、下一曲、音量调节等功能
		(2)、按键一对一控制模式	分别由不同的按键控制 5 段语音播放
		(3)、随机播放控制模式	任何按键均无效，上电默认播放第一曲，随后的为随机形式
		(4)、MCU 控制模式	DSA 控制模式，能跟以上三种控制模式并存
3	上电方式	(1)、上电播放语音	通电后无需任何操作直接播放第一曲语音
		(2)、上电不播放语音	通电后无任何动作
4	存储区选择	(1)、从公共区开始播放	默认上电操作区域为公共区
		(2)、从隐藏区开始播放	默认上电操作区域为隐藏区
5	存储区循环	(1)、存储区分别循环播放	全部语音循环播放时有效，可在当前操作的储存区循环
		(2)、所有存储区循环播放	全部语音循环播放时有效，能在公共区和隐藏区之间循环

在序号 1～5 的选项中，同一个序号里只能设置一个功能。序号 2 中，选项(1)(2)(3)只能设置其中的一种控制模式，选项(4)在任何控制模式下都存在。

6、公共区/隐藏区音乐描述

公共区音乐，指的是可以通过 USB 线直接下载到 NAND-Flash 根目录的 MP3 音乐。

隐藏区音乐，指的是需要订制的，由我司拷贝到 NAND-Flash，并且客户端无法修改及格式化的 MP3 音乐。

7、控制模式

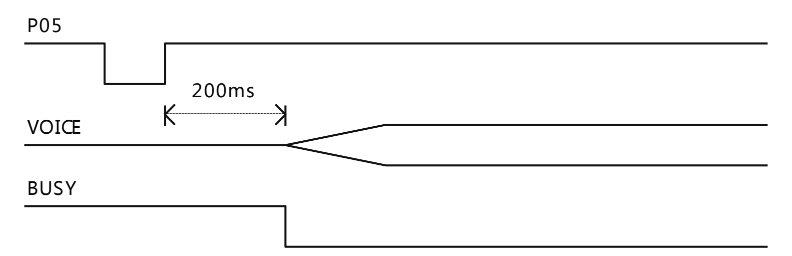
WT8601M02 支持 MP3 控制模式、按键一对一控制模式、随机播放控制模式、MCU 控制模式等四种控制模式，控制模式可通过 MCU 发码更换。

7.1、MP3 控制模式

在 MP3 控制模式下，I/O P01～P05 保持 10ms 的低电平，就能触发相关的功能。各 I/O 所对应的功能如下。P05 短按 1 秒为播放/暂停功能，长按 3 秒为停止功能。P00 为输出口，语音播放过程中为低电平，语音停止时为高电平。

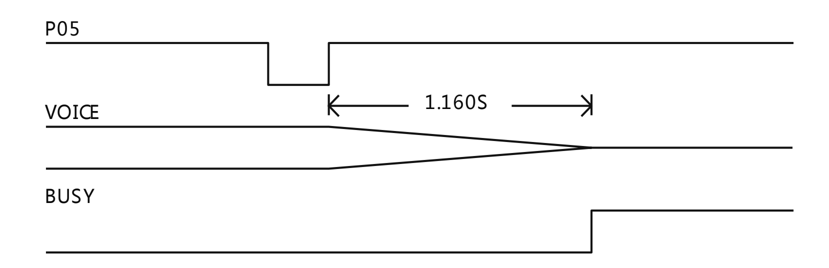
I/O	P00	P01	P02	P03	P04	P05
功能	BUSY	音量+	音量-	上一曲	下一曲	播放/暂停/停止

播放操作



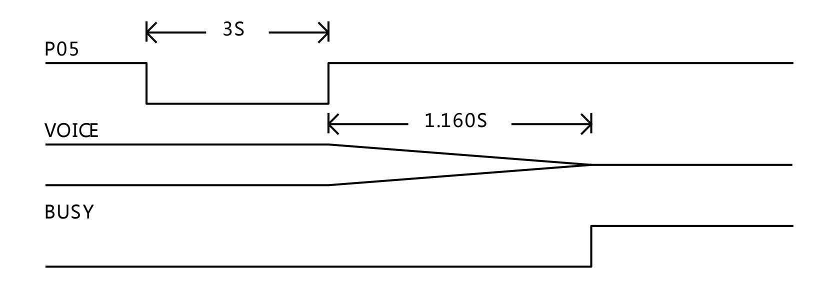
在语音停止状态，用 10ms～1S 的低电平触发 P05，就能触发播放语音。触发后 200ms 开始播放语音，同时 BUSY 转为低电平。语音以渐进放大的功能播放。

暂停操作



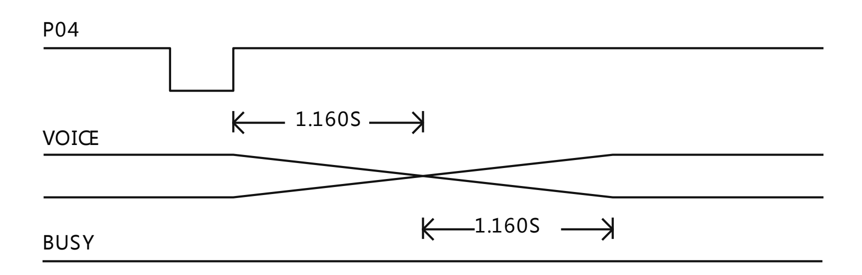
在语音播放状态，用10ms～1S 的低电平触发 P05，就能暂停当前的语音。触发后语音开始逐渐减小，1.160S 后完全停止播放，同时 BUSY 信号转为高电平。

停止操作



在语音播放状态，P05保持3S 的低电平，就能停止播放当前的语音。触发后语音开始逐渐减小，1.160S 后完全停止播放，同时 BUSY 信号转为高电平。

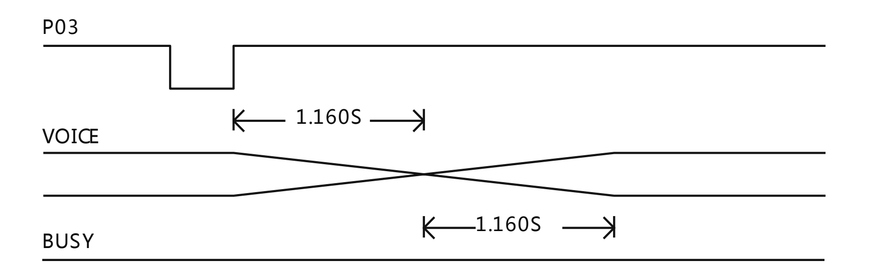
下一曲操作



在语音播放状态，用10ms～1S 的低电平触发 P04，当前语音声音逐渐减小1.160S 后停止播放，切换到下一曲开始播放，语音播

放时声音逐渐增大。在语音播放过程中切换到下一曲语音，BUSY 一直为低电平。

上一曲操作



在语音播放状态，用10ms～1S 的低电平触发 P03，当前语音声音逐渐减小1.160S 后停止播放，切换到上一曲开始播放，语音播放时声音逐渐增大。在语音播放过程中切换到上一曲语音，BUSY 一直为低电平。

7.2、按键一对一控制模式

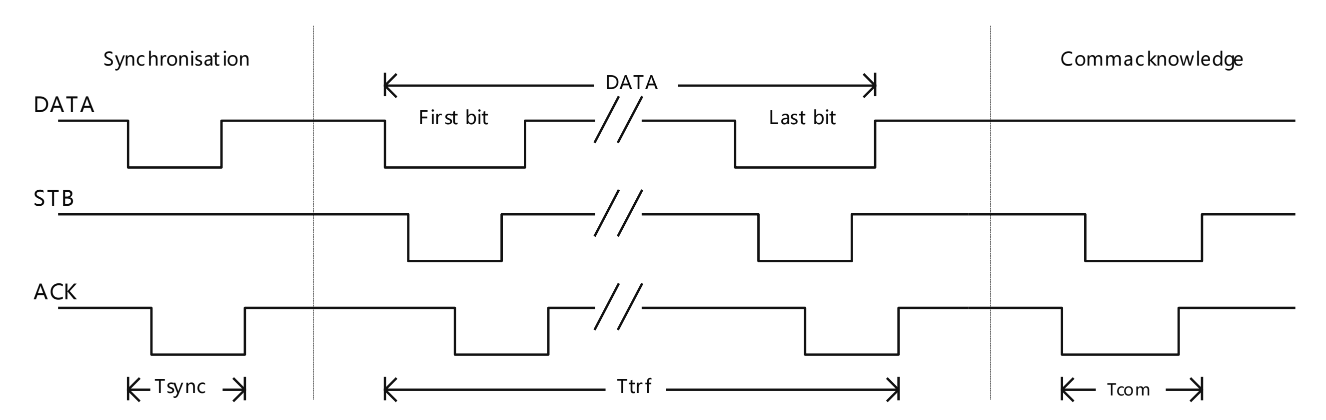
按键一对一控制模式下，WT8601M02 最多只能播放 5 首音乐，且一个 I/O 对应一段音乐。I/O P01～P05 保持 10ms 的低电平，就能触发相关的功能。P00 为输出口，语音播放过程中为低电平，语音停止时为高电平。

I/O	P00	P01	P02	P03	P04	P05
功能	BUSY	第一首	第二首	第三首	第四首	第五首
对应文件名	无	第一首语音	第二首语音	第三首语音	第四首语音	第五首语音

7.3、MCU 控制模式

MCU 控制模式通过 DSA_DATA、DSA_ACK、DSA_STB 三个端口来控制 WT8601M02 工作。本协议以标准 DSA 模式做为基础修改，占用系统资源少，对时间没有严格要求。

7.3.1、控制时序



先从 DATA 发任一数据给 WT8601M02，当 WT8601M02 收到数据后，通过 ACK 给 MCU 发返回信息，检测到 ACK 为低低电平后把 DATA 拉高，当 ACK 为高电平后，发送 First 后再发送 STB，检测 ACK 为低电平后把 STB 拉高，继而把 DATA 也拉高，只有检测到 ACK 为高电平后才能继续发送下一位数据。用同样的方式发送完 Last bit ,并检测到 ACK 为高电平后 ,由 MCU 往 WT8601M02 发送 ACK，随后按顺势发送 STB，拉高 ACK 和 STB，完成此操作后 WT8601M02 才能确定之前发的数据为有效数据。

如果 Tsync、Ttrf、Tcom 中的任何一个时间超出 250ms，则被 WT8601M02 判断为失败数据。数据先发高位再发低位。

7.3.2、隐藏区语音命令码

公共区/隐藏区音乐切换播放

起始码	长度	命令	结束码
7E	02	B0	7E

发送该指令，可以在公共区及隐藏区之间切换。

7.3.3、公共区/隐藏区音乐命令码

公共区和隐藏区的播放命令是一致的，当前操作是在哪个区域，下列命令就对该区域的操作有效。

语音播放

起始码	长度	命令	固定数据	曲目	结束码
7E	04	A5	00	01	7E

发送数据时，改变曲目就可以点播不同的音乐。曲目也为 16 进制数据。发送 7E 04 A5 00 01 7E 为播放第一曲音乐。

语音暂停

起始码	长度	命令	结束码
7E	02	A1	7E

第一次发送该指令，则暂停播放音乐，再次发送该数据，则从暂停处继续播放音乐。

语音停止

起始码	长度	命令	结束码
7E	02	A3	7E

发送该指令，停止播放当前正在播放的音乐。

音量控制

起始码	长度	命令	音量等级	结束码
7E	03	A4	31	7E

音量等级共有 32 级，分别为 00～31，其中 00 为静音，31 级为最大音量，上电默认为 16 级。如果发送 8 级音量，则发送指令为 7E 03 A4 08 7E

上一曲

起始码	长度	命令	结束码
7E	02	A7	7E

该指令能够触发播放上一曲音乐，在播放最后一曲音乐时，发送该指令可触发播放第一曲音乐。

下一曲

起始码	长度	命令	结束码
7E	02	A6	7E

该指令能够触发播放下一曲音乐，在播放第一曲音乐时，发送该指令可触发播放最后一曲音乐。

7.3.4、读取返回信息

读取公共区语音总数

发送			接收		
起始码	长度	命令	固定数据	曲目	结束码
7E	04	C2	00	XX	7E

MCU 发送 7E 04 C2 后，WT8601M02 返回 00 XX 7E，其中 XX 为公共区语音曲目总数。

读取隐藏区语音总数

发送			接收		
起始码	长度	命令	固定数据	曲目	结束码
7E	04	C3	00	XX	7E

MCU 发送 7E 04 C3 后，WT8601M02 返回 00 XX 7E，其中 XX 为隐藏区语音曲目总数。

读取当前设置音量

发送			接收	
起始码	长度	命令	音量	结束码
7E	03	C5	XX	7E

MCU 发送 7E 03 C5 后，WT8601M02 返回 XX 7E，其中 XX 为当前音量等级。

读取当前播放语音索引

索引：指语音在存储设备中的排列序号，跟语音文件复制到存储设置中的顺序对应。

发送			接收			
起始码	长度	命令	播放区域	固定数据	曲目	结束码
7E	05	C1	XX	00	XX	7E

MCU 发送 7E 05 C1 后，WT8601M02 返回 XX 00 XX 7E，其中返回的信息中播放区域的 00 表示公共区，01 表示隐藏区。曲目信息为当前公共区/隐藏区正在播放的语音索引。

7.4、功能设置命令码

起始码	长度	命令	功能	结束码
7E	03	AA	XX	7E

MCU 发送 7E 03 AA XX 7E，就能设置模块的控制功能，其中 XX 为功能设置代码，具体功能如下表所示。

序号	XX 数据		功能说明	备注
1	D7	0	默认开机音量	设置开机音量暂时不可用
		1	设置开机音量	
2	D6	0	全部语音循环播放	D5 设置为 1 时，D6 的设置无效。
		1	单曲语音循环播放	
3	D5	0	D6 设置有效	
		1	单曲语音播放停止	
4	D4	0	MP3 控制模式	D4、D3 组合使用，DSA 控制模式在任何模式下都存在。
		1	按键一对一控制模式	
5	D3	2	随机模式	
		---	DSA 控制模式	
6	D2	0	存储区分别循环播放	
		1	所有存储区循环播放	
7	D1	0	从隐藏区开始播放	
		1	从公共区开始播放	
8	D0	0	开机不播放语音	
		1	开机播放语音	

D4、D3 部分说明

控制模式	D4	D3
MP3 控制模式	0	0
按键一对一控制模式	0	1
随机控制模式	1	0

8、控制程序范例

8.1、汇编程序

;**

;项目名:WT8601M02 模块　DSA 通信测试程序

;功能要求:

;　　1) 通过 RS232 和 PC 进行通信(暂无)

;　　2) 通过 DSA 通信协议和 WT8601M02 模块通信

;　　3) 实现简易按键控制方式

;硬件配置:

```
;       1) MCU 型号: AT89C2051
;       2)  外接晶振频率为:11.0592MHz
;       3) I/O  定义:

        ACK         EQU P3.2      ;ACK 反馈信号引脚（通信口不要加其他上拉等，3V 供电）
        STB         EQU P3.3      ;STB 发送引脚     （通信口不要加其他上拉等，3V 供电）
        DAT         EQU P3.4      ;数据引脚          （通信口不要加其他上拉等，3V 供电）
FLAG_10MS       BIT 20H.0     ;10MS 定时标志
comm_flag       bit 20h.1     ;命令发送标志
comm_ok         bit 20h.2     ;dsa 命令成功标志
timeout_flag bit 20h.3     ;时间超出标志
   STARTN       EQU 30H       ;DSA 数据暂存空间
      NUM1      EQU 31H
      NUM2      EQU 32H
      NUM3      EQU 33H
      NUM4      EQU 34H
      NUM5      EQU 35H
      NUM6      EQU 36H
      NUM7      EQU 37H
      ENDN      EQU 38H       ;DSA 数据暂存空间结束

      VOLN      EQU 40H       ;音量级数暂存
dsa_timeout   EQU 41H       ;dsa 时间限制计数器
P0_IN_REG     EQU 60H       ;P0 口按键比较值
    DUMMY       EQU 61H       ;扫描暂存值
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
           ORG       0000H
           LJMP      START
           ORG       000BH
           LJMP      TIME0
;          org       0023h
;     LJMP      SERIAL
           ORG       0030H
```

;;;;;;;;;;;; 主 程 序 ;;;;;;;;;;;;;;;;

```
START:
        NOP
        NOP
        MOV     R0,#STARTN
        MOV     R5,#32
CLEAR:MOV     @R0,#0
        INC     R0
        DJNZ    R5,CLEAR        ;清零寄存器
        MOV     VOLN,#16
        MOV     P0_IN_REG,#0FFH
        MOV     20H, #01H       ;初始化
        mov     SCON,#50H
        MOV     TMOD,#21H
        MOV     TH0, #0D8H
        MOV     TL0, #0F0H      ;TIME0   10MS 定时
        mov     TH1, #0F3H
        mov     TL1, #0F3H      ;TIME1   初始化波特率(2400)
        SETB    REN             ;允许串口接收
        SETB    ES              ;开串口中断
        SETB    TR0
        SETB    TR1
        SETB    ET0
        MOV     P3,#0FFH
        SETB    EA
MAIN:
        LCALL   SCAN_KEY                ;按键扫描
        LCALL   dsa_data_transmit  ;数据处理
        LJMP    MAIN

;;;;;;;;;;;;;串口中断程序（暂时没用）;;;;;;;;;;;;;
SERIAL:
        JNB     RI,SEND_RET
        CLR     RI
```

```
        MOV      A,SBUF
        RETI
SEND_RET:
        CLR      TI
SERIAL_END:
        RETI

;;;;;;;;;;;;;;10MS 定时中断;;;;;;;;;;;;;;;;;;;
TIME0:
        PUSH     ACC
        PUSH     PSW
        CLR      TR0
        MOV      TH0,#0D8H
        MOV      TL0,#0F0H
        SETB     FLAG_10MS
        dec      dsa_timeout
        MOV      A,dsa_timeout
        CJNE     A,#0,TIME00
        SETB     timeout_flag
TIME00:
        SETB     TR0
        POP      PSW
        POP      ACC
        RETI

;;;;;;;;;;;;;;;;;;;;数据处理程序;;;;;;;;;;;;;;;
dsa_data_transmit:
        jb       comm_flag,transmit
            ret
transmit:
        mov      r0,#startn
        lcall    dsa_syn_start
        jnb      comm_ok,dsa_data_reset
```

```
        mov     a,@r0    ;startn
        lcall   dsa_send_byte
        jnb     comm_ok,dsa_data_reset

        inc     r0
        mov     a,@r0    ;num1
        mov     r4,a     ;存放字节长度
        lcall   dsa_send_byte
        jnb     comm_ok,dsa_data_reset

transmit_loop:
        inc     r0
        mov     a,@r0    ;num1
        lcall   dsa_send_byte
        jnb     comm_ok,dsa_data_reset
        DJNZ    R4,transmit_loop

        lcall   dsa_comm_acknowledge
        jnb     comm_ok,dsa_data_reset
dsa_data_reset:
        clr     comm_flag       ;数据发送接收
        SETB    STB
        SETB    ack
        SETB    dat
        ret

;;;;;;;;;;;;;;按键扫描程序;;;;;;;;;;;;;;;;;;;;;;;;
SCAN_KEY:
        JB      FLAG_10MS,KEY
                RET
      KEY:
                CLR     FLAG_10MS
                MOV     A,P1
                mov     DUMMY,a
```

```
            XRL         A,P0_IN_REG             ;第一次初始化为 0FFH
            ANL         A,P0_IN_REG     ;判断为下降沿确定为按键按下
            MOV         P0_IN_REG,DUMMY;存 P0 口本次扫描的值
            CJNE    A,#0H,K1
        RET
    K1:
            CJNE    a,#01h,K2
            LCALL   music_one       ;播放第一首
            RET
      K2:
            CJNE    a,#02h,K3
            LCALL   music_pause     ;播放/暂停
            RET
      K3:
            CJNE    a,#04h,K4
            LCALL   music_stop      ;语音停止
            RET
      K4:
            CJNE    a,#10h,K5
            LCALL   music_up        ;上一曲
            RET
      K5:
            CJNE    a,#20h,K6
            LCALL   music_down      ;下一曲
            RET
    K6:
            CJNE    a,#40h,K7
            LCALL   vol_inc         ;音量+
            RET
    K7:
        CJNE      a,#80h,KEY_END
        LCALL   vol_dec           ;音量-
            RET
KEY_END:
```

```
            RET
;=======================================================================
;功能描述: 赋值对应的控制命令
;=======================================================================
music_one:
        mov     startn,#7eh
        mov     num1,#04h
        mov     num2,#0a5h
        mov     num3,#00h
        mov     num4,#01h
        mov     num5,#7eh
        mov     num6,#00h
        mov     num7,#00h
        mov     endn,#00h
        SETB    comm_flag
        ret
music_pause:
        mov     startn,#7eh
        mov     num1,#02h
        mov     num2,#0a1h
        mov     num3,#7eh
        mov     num4,#00h
        mov     num5,#00h
        mov     num6,#00h
        mov     num7,#00h
        mov     endn,#00h
        SETB    comm_flag
        ret
music_stop:
        mov     startn,#7eh
        mov     num1,#02h
        mov     num2,#0a3h
        mov     num3,#7eh
        mov     num4,#00h
```

```
        mov     num5,#00h
        mov     num6,#00h
        mov     num7,#00h
        mov     endn,#00h
        SETB    comm_flag
        ret
music_up:
        mov     startn,#7eh
        mov     num1,#02h
        mov     num2,#0a7h
        mov     num3,#7eh
        mov     num4,#00h
        mov     num5,#00h
        mov     num6,#00h
        mov     num7,#00h
        mov     endn,#00h
        SETB    comm_flag
        ret
music_down:
        mov     startn,#7eh
        mov     num1,#02h
        mov     num2,#0a6h
        mov     num3,#7eh
        mov     num4,#00h
        mov     num5,#00h
        mov     num6,#00h
        mov     num7,#00h
        mov     endn,#00h
        SETB    comm_flag
        ret
vol_inc:
        mov     startn,#7eh
        mov     num1,#03h
        mov     num2,#0a4h
```

```
        mov     num4,#7eh
        mov     num5,#00h
        mov     num6,#00h
        mov     num7,#00h
        mov     endn,#00h
        mov     a,voln
        CJNE     a,#31,inc_x
inc_x:
        jnc     inc_end
        inc     voln
        mov     num3,voln
inc_end:
        SETB    comm_flag
        ret
vol_dec:
        mov     startn,#7eh
        mov     num1,#03h
        mov     num2,#0a4h
        mov     num4,#7eh
        mov     num5,#00h
        mov     num6,#00h
        mov     num7,#00h
        mov     endn,#00h
        mov     a,voln
        CJNE     a,#0,dec_x
dec_end:
        SETB    comm_flag
        ret
dec_x:
        dec     voln
        mov     num3,voln
        SETB    comm_flag
        ret
```

```
;=====================================================================
;函数名   : dsa_syn_start
;功能描述: 启动 DSA 总线，产生同步信号(250ms 超时)
;输入参数:
;输出参数: 启动是否成功标志，成功 comm_ok=1，失败 comm_ok=0
;=====================================================================
dsa_syn_start:
        mov     dsa_timeout,#25
        CLR     timeout_flag
        SETB    STB
        CLR     DAT

start_ackl:
        JB      timeout_flag,start_err
        JB      ACK,start_ackl    ;等待 ack 为 0

        SETB    DAT
start_ackh:
        JB      timeout_flag,start_err
        JNB     ACK,start_ackh   ;等待 ack 为 1
start_ok:
        SETB    comm_ok
        ret
start_err:
        CLR     comm_ok
        ret

;=====================================================================
;函数名   : dsa_comm_acknowledge
;功能描述: 停止 DSA 总线，产生应答信号(250ms 超时)
;输入参数:
;输出参数: 停止总线是否成功标志，成功 comm_ok=1，失败 comm_ok=0
;=====================================================================
dsa_comm_acknowledge:
```

```
        mov       dsa_timeout,#25
        CLR       timeout_flag
        SETB      STB
        CLR       ack

stop_ackl:
        JB        timeout_flag,stop_err
        JB        STB,stop_ackl       ;等待 STB 为 0

        SETB      ack
stop_ackh:
        JB        timeout_flag,stop_err
        JNB       STB,stop_ackh       ;等待 STB 为 1
stop_ok:
        SETB      comm_ok
        ret
stop_err:
        CLR       comm_ok
        ret

;=====================================================================
;函数名   : dsa_send_byte
;功能描述: DSA 总线写一字节数据(250ms 超时)
;输入参数: acc
;输出参数: 发送 DSA 数据是否成功,成功 comm_ok=1，失败 comm_ok=0
;=====================================================================
dsa_send_byte:
        mov       dsa_timeout,#25
        CLR       timeout_flag
        MOV       R5,#08H
send_loop:
        RLC       A
        MOV       DAT,C               ;先发送高位
```

```
        CLR      STB
send_ackl:
        JB       timeout_flag,send_err
        JB       ACK,send_ackl    ;等待 ack 为 0

        SETB     STB
send_ackh:
        JB       timeout_flag,send_err
        JNB      ACK,send_ackh  ;等待 ack 为 1
        DJNZ     R5,send_loop
send_ok:
        SETB     DAT
        SETB     comm_ok
        ret
send_err:
        CLR      comm_ok
        ret
            END
```

8.2、C 语言程序

```
#include <iom8v.h>
#include <macros.h>
#include "define.h"
#include "dsa_ctl.h"

extern uint8     uc_dsa_data;
extern uint8     uc_dsa_timeout;

INT8U dsa_bus_data_transmit(INT8U * dsa_tx_buff);

void delay_x10us(INT8U times)
{
      while(--times);
```

```
}

//=================================================================
//函数名  : dsa_bus_syn_start
//功能描述: 启动 DSA 总线，产生同步信号
//输入参数: 无
//输出参数: 启动是否成功标志
//=================================================================
INT8U dsa_bus_syn_start()
{
      P_DSA_DATA_LOW();
       DSA_DATA_OUTPUT();
      P_DSA_STB_HIGHT();
       DSA_STB_OUTPUT();
       DSA_ACK_INPUT();
      uc_dsa_timeout = DSA_TIMEOUT;                    //同步 250ms 则超时退出]
      while(P_DSA_GET_ACK_PIN() && (uc_dsa_timeout>0));
       if(uc_dsa_timeout == 0) return(FAILURE);
       P_DSA_DATA_HIGHT();
      while((!P_DSA_GET_ACK_PIN()) && (uc_dsa_timeout>0));
      if(uc_dsa_timeout == 0) return(FAILURE);
      return(SUCCESS);
}

//=================================================================
//函数名  : dsa_bus_ack_write
//功能描述: 停止 DSA 总线，产生应答信号
//输入参数: 无
//输出参数: 停止总线是否成功标志
//=================================================================
INT8U dsa_bus_ack_write(void)
{                                                        //250ms 超时
       P_DSA_ACK_LOW();                                  //ACK 拉低，等待 STB 拉低
       DSA_ACK_OUTPUT();
```

```
	uc_dsa_timeout = DSA_TIMEOUT/10;
	DSA_STB_INPUT();
	DSA_DATA_INPUT();
	while(P_DSA_GET_STB_PIN() && (uc_dsa_timeout>0));
	if(uc_dsa_timeout == 0) return(FAILURE);
	P_DSA_ACK_HIGHT();					//ACK 拉高，等待 STB 拉高
	delay_x10us(10);
	//	while((!P_DSA_GET_STB_PIN()) && (uc_dsa_timeout>0));
	//	if(uc_dsa_timeout == 0) return(FAILURE);
	return(SUCCESS);
}

//==================================================================
//函数名	: dsa_bus_ack_read
//功能描述: 读 DSA 应答信号
//输入参数: 无
//输出参数: 读 DSA 应答信号是否成功标志
//==================================================================
INT8U dsa_bus_ack_read(void)				//250ms 超时
{
	P_DSA_ACK_HIGHT();
	DSA_STB_INPUT();
	DSA_ACK_OUTPUT();
	uc_dsa_timeout = DSA_TIMEOUT/10;
	while(P_DSA_GET_STB_PIN() && (uc_dsa_timeout>0));
	if(uc_dsa_timeout == 0) return(FAILURE);
//	DSA_STB_OUTPUT();
	P_DSA_ACK_LOW();					//pull down DSA_ACK
	while((!P_DSA_GET_STB_PIN()) && (uc_dsa_timeout>0));
	if(uc_dsa_timeout == 0) return(FAILURE);
	P_DSA_ACK_HIGHT();
	return(SUCCESS);
}
```

```
//=================================================================
//函数名   : dsa_bus_byte_write
//功能描述: DSA 总线写一字节数据(125ms 超时)
//输入参数: 待发送的 DSA 数据
//输出参数: 发送 DSA 数据是否成功
//=================================================================
INT8U dsa_bus_byte_write(INT8U temp_data)
{
    INT8U x;
    uc_dsa_timeout = DSA_TIMEOUT/10;
    for(x = 0; x < 8; x++)
    {
        if(temp_data & 0x80)
            P_DSA_DATA_HIGHT();
        else
            P_DSA_DATA_LOW();

        temp_data <<= 0x01;
        P_DSA_STB_LOW();                         //DSA_STB 拉低,等待 ACK 拉低应答
        DSA_ACK_INPUT();
        while(P_DSA_GET_ACK_PIN() && (uc_dsa_timeout>0));
        if(uc_dsa_timeout == 0) return(FAILURE);
        P_DSA_STB_HIGHT();                       //拉高 DSA_STB,等待 ACK 拉高
        while((!P_DSA_GET_ACK_PIN()) && (uc_dsa_timeout>0));
        if(uc_dsa_timeout == 0) return(FAILURE);
    }
    return(SUCCESS);
}

//=================================================================
//函数名   : dsa_bus_byte_read
//功能描述: DSA 总线读一字节数据(125ms 超时)
//输入参数: NONE
//输出参数: 1.读数据是否成功标志
```

```
//                2.读出的数据(uc_dsa_data)
//=================================================================
INT8U dsa_bus_byte_read(void)
{                                                              //125ms 超时
      INT8U x;
      uc_dsa_data = 0;
      uc_dsa_timeout = DSA_TIMEOUT/10;
      for(x = 0; x < 8; x++)
      {
           DSA_ACK_INPUT();                                   //等待 DSA_ACK 拉低
            while(P_DSA_GET_ACK_PIN() && (uc_dsa_timeout>0));
            if(uc_dsa_timeout == 0) return(FAILURE);
            uc_dsa_data <<= 1;
            if(P_DSA_GET_DATA_PIN())
                  uc_dsa_data |= 0x01;
            P_DSA_STB_LOW();                                  //DSA_STB 拉低,等待 ACK 拉高
            while(!P_DSA_GET_ACK_PIN() && (uc_dsa_timeout>0));
            if(uc_dsa_timeout == 0) return(FAILURE);
            P_DSA_STB_HIGHT();
     }
      return(SUCCESS);
}

//=================================================================
//函数名  : dsa_bus_communicate
//功能描述:
//输入参数:
//输出参数: NONE
//=================================================================
INT8U dsa_bus_communicate(INT8U *dsa_data_buff)
{
      INT8U dsa_point,dsa_length;

      if(dsa_bus_syn_start() != SUCCESS) goto dsa_bus_reset;
```

```
    if(dsa_bus_byte_write(dsa_data_buff[0]) != SUCCESS) goto dsa_bus_reset;
    if(dsa_bus_byte_write(dsa_data_buff[1]) != SUCCESS) goto dsa_bus_reset;
    if(dsa_bus_byte_write(dsa_data_buff[2]) != SUCCESS) goto dsa_bus_reset;

    dsa_point = 0x03;
    dsa_length = dsa_data_buff[1] - 1;
    if((dsa_data_buff[2]&0xf0) != 0xC0)
    {
        while(dsa_length--)
        {
            if(dsa_bus_byte_write(dsa_data_buff[dsa_point++]) != SUCCESS)
                goto dsa_bus_reset;
        }
        if(dsa_bus_ack_write() != SUCCESS) goto dsa_bus_reset;
    }
    else
    {
        DSA_DATA_INPUT();
        while(dsa_length--)
        {
            if(dsa_bus_byte_read() != SUCCESS) goto dsa_bus_reset;
            dsa_data_buff[dsa_point++] = uc_dsa_data;
        }
        if(uc_dsa_data != 0x7E) goto dsa_bus_reset;
        if(dsa_bus_ack_read() != SUCCESS) goto dsa_bus_reset;
    }
    DSA_DATA_INPUT();
    DSA_STB_INPUT();
    DSA_ACK_INPUT();
    return(SUCCESS);
    //------------------------------------------------------------------
dsa_bus_reset:                                  //DSA 写数据通讯超时推出
    DSA_DATA_INPUT();
    DSA_STB_INPUT();
```

```
    DSA_ACK_INPUT();
    return(FAILURE);
}

//========================================================================
//函数名   : dsa_bus_data_transmit
//功能描述: 发送通讯命令,如果不成功,则重复发送 3 次,超过 3 次则不再重新发送
//输入参数: dsa_tx_buff
//输出参数: NONE
//========================================================================
INT8U dsa_bus_data_transmit(INT8U * dsa_tx_buff)
{
    INT8U cnt;

    for(cnt = 0; cnt < 3; cnt++)
    {
        if(dsa_bus_communicate(dsa_tx_buff) == SUCCESS) return(SUCCESS);
    }
    return(FAILURE);                                //3 次通讯不成功，退出
}

//========================================================================
```

9、NAND-FLASH 文件内容

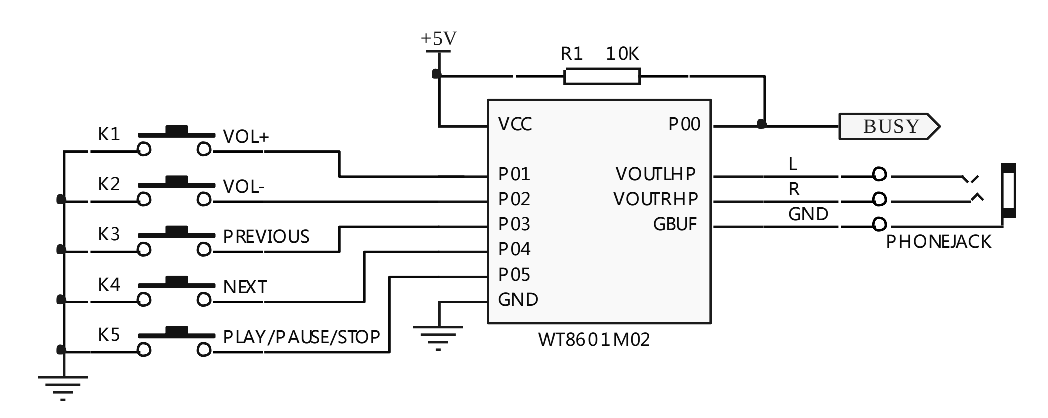
WT8601M02 以音频文件先后存放的顺序来区分曲目顺序，对歌曲名不做任何要求。为了更好的管理 NAND-Flash 中的音频文件内容，可以才用重命名的方式。

通过 USB 线把 WT8601M02 连接到电脑，先在 PC 端对 MP3 文件进行重命名操作，如“加州旅馆.mp3”，重命名为“0001 加州旅馆.mp3”，并把要拷贝到 WT8601M02 的 MP3 音频文件复制到一个空文件夹里，再按键盘“Ctrl+A”全选，接着按“Ctrl+C”复制所有 MP3 音频，回到 WT8601M02 在 PC 端的存储盘，按“Ctrl+V”粘贴，把 MP3 音频直接复制到 NAND-FLASH，这样 WT8601M02 就把该曲目定义为 01 地址的音频 ,发送 7E 04 A5 00 01 7E 就能点播该音乐。如果没有重命名 mp3 文件 ,WT8601M02 会按照文件存放的顺序记录曲目，虽然可以播放，但是不便于控制操作。选中、复制、粘贴操作只能通过键盘来完成，请勿使用鼠标操作，否则容易出错。

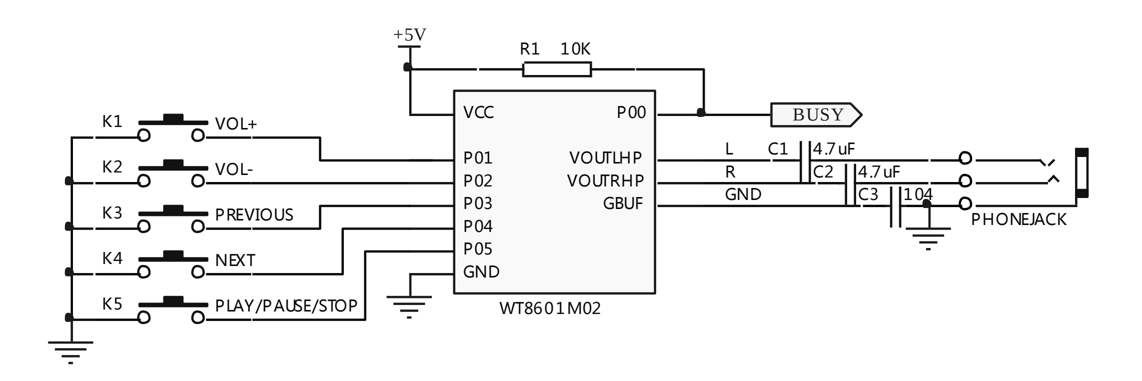
10、应用电路

10.1、MP3 控制模式应用电路

外接耳塞模式

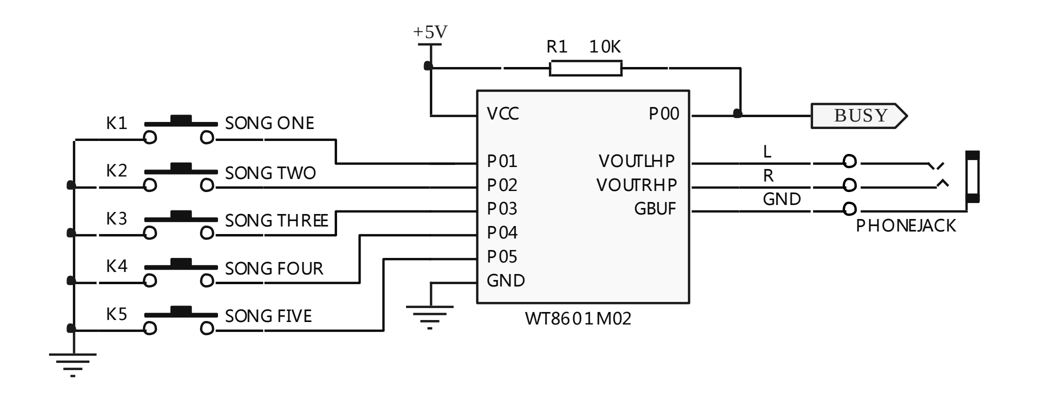


外接功放模式

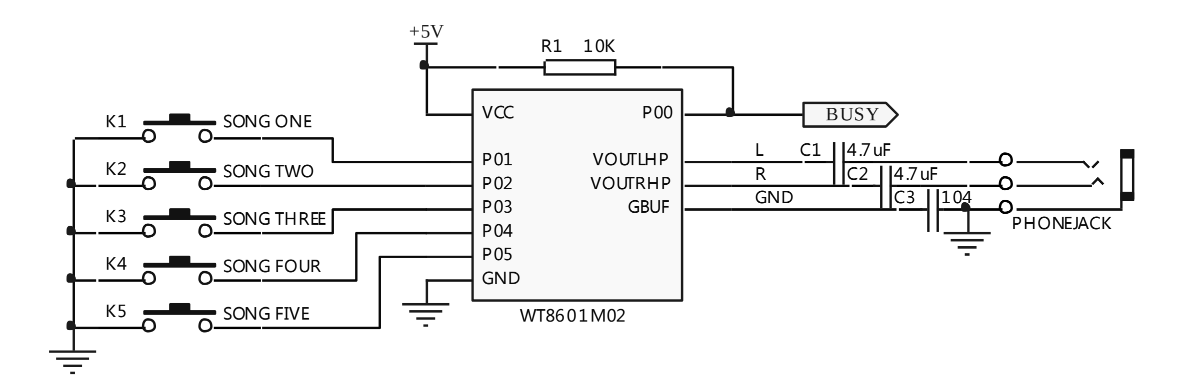


10.2、按键一对一控制模式应用电路

外接耳塞模式



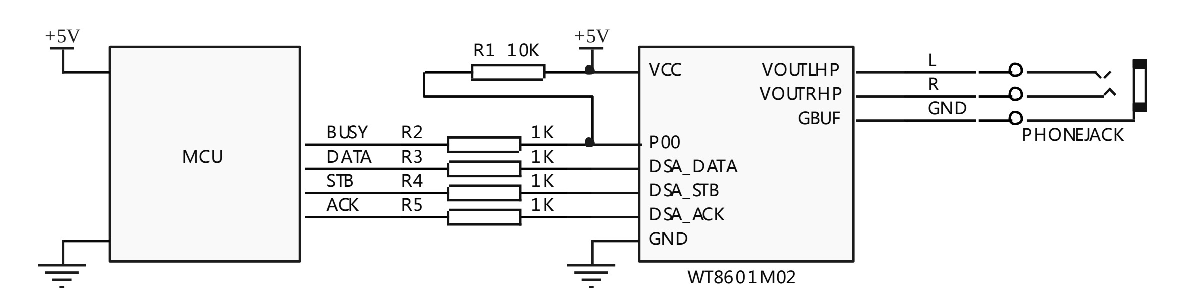
外接功放模式



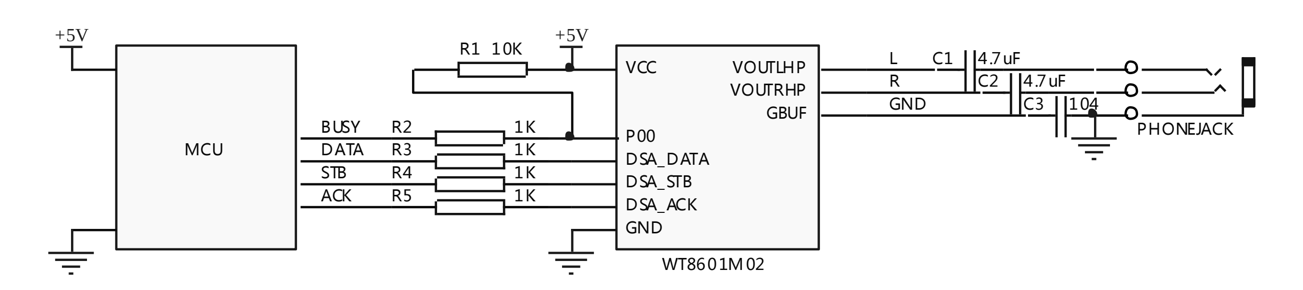
10.3、MCU 控制模式应用电路

在 MCU 控制模式中，P00 为 BUSY 输出端，语音停止播放时，BUSY 为高电平，语音播放过程中，BUSY 为低电平。可以由 MCU 来检测 WT8601M02 的语音播放状态。如 MCU 为 3V 供电，则不用接 R2、R3、R4、R5。

外接耳塞模式

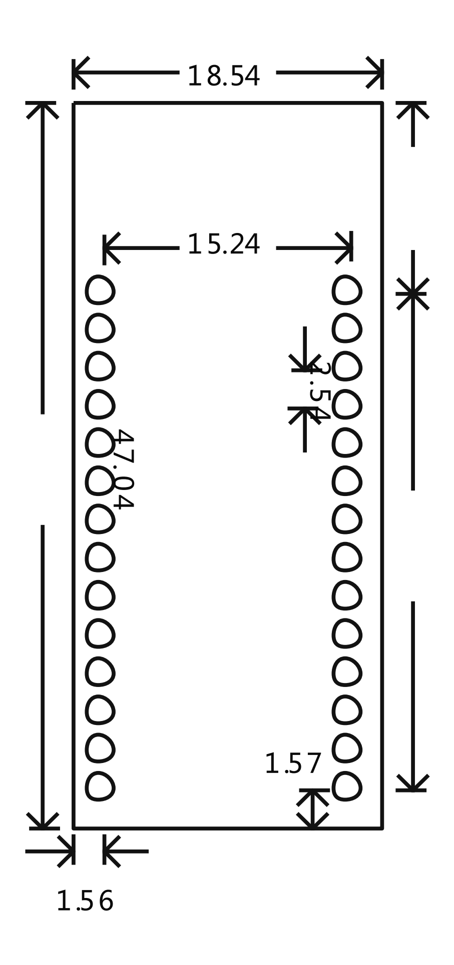


外接功放模式



11、封装尺寸图

单位：mm



12、历史版本记录

版本	日期	描述
V1.0	2009-9-24	原始版本
V1.1	2009-10-12	修正电路图部分
V1.2	2009-10-28	修改部分描述
V1.3	2009-11-13	修正DSA控制程序，完善DSA命令描述，修改应用电路

广州唯创电子有限公司（原广州唯创科技有限公司）1999 年创立于广州市天河区，是一家集语音芯片研发、语音产品方案设计、语音产品生产、语音编辑上位机软件开发的高新技术公司。业务范围涉及汽车电子、多媒体、家居防盗、通信、家电、医疗器械、工业自动化控制、玩具及互动消费类产品等领域。团队有着卓越的 IC 软、硬件开发实力和设计经验，秉持着「积极创新、勇于开拓、满足顾客、团队合作」的理念，力争打造“语音业界”的领导品牌。

唯创主要生产 WTV 系列语音芯片、WTR 可录音系列语音芯片、WT588D 语音芯片、WTB 系列语音芯片、WTM 系列高音质语音应用模块、WTF 系列的高性价比长时间播放模块，及特约代理的 APLUS 系列语音芯片、ISD 全系列可录放语音芯片等。率先提供最完备、多元化的客需解决方案，节约研发成本，缩短研发周期，使产品在最短的时间内成熟上市。在汽车电子及特种车领域，自主研发的公交车报站器在国内有着很好的市场口碑，为叉车使用安全而开发的叉车超速报警器是国内第一家研发此类产品并大量生产的企业。

唯创坚持“以人为本，不断进行核心技术创新，优良的售后技术跟踪服务”的经营策略，使得唯创能傲立于语音产品行业。WTV 系列语音芯片、WTR 可录音系列语音芯片、WTM 系列高音质语音应用模块、WTF 系列的高性价比长时间播放模块等都是唯创的自主品牌，具有很强的市场竞争优势。产品、模块、编辑软件等的人性化设计，使得客户的使用更方便。于 2006 年新成立的北京唯创虹泰分公司主要以销售完整的方案及成熟产品为宗旨，以便于为国内北方客户提供更好的服务。

唯创持续在研发与技术升级领域大力投资，每年平均提拨超过 20%的营业额作为研发经费，在我们的研发团队中，有超过90%员工钻研技术及产品发展。并与同行业大厂合作，勇于迈出下一个高峰。

总公司名称：广州市唯创电子有限公司
电话：020-85638660　85638557　85638319　　传真：020-85638637
业务销售 E-mail：sos@1999c.com　　网址：http://www.w1999c.com
地址：广东省广州市天河区棠东东路 25 号 5 楼

分公司名称：北京唯创虹泰科技有限公司
电话：010-89756745　　传真：010-89750195
E-mail：BHL8664@163.com　　网址：http://www.w1999c.com
地址：北京昌平区立汤路 186 号龙德紫金 3#902 室

广州唯创电子有限公司深圳办事处
电话：0755-83044339　　传真：0755-83044339
地址：深圳福田区福华路 110 号广业大厦东座 22G 室